UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/767,340	01/30/2004	Bryan R. Goring	T8467901US	5223

92030 7590 11/15/2010

Gowling Lafleur Henderson LLP
Suite 1600 1 First Canadian Place 100 King Street West
Toronto, ON M5X1G5
CANADA

EXAMINER	
DAO, THUY CHAN	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
11/15/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/767,340	GORING ET AL.	
	Examiner	Art Unit	
	Thuy Dao	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 September 2010*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-8,10-28 and 30-42* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-8,10-28 and 30-42* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 January 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date ____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to Applicant' reply filed on September 3, 2010.

2. Claims 1-8, 10-28, and 30-42 have been examined.

Claim Objections

3. Claims are objected to because of minor informalities.

Claim 1 recites "the content on the terminal" and is considered to read as - -the content of the application- -.

Claim 8 ends with a comma.

Claim 26 ends with a semi-colon.

Response to Arguments

4. Applicant's arguments have been fully considered. After further consideration, a new ground of rejection is made as set forth in details below.

Claim Rejections – 35 USC §112

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. Claims 1-8, 10-28, and 30-40 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 1 recites the limitation "the specified content type". There is insufficient antecedent basis for this limitation in the claim.

Claim 1 recites "the provisioning instructions embedded with content of the application". However, dependent claim 10 further recites "the provisioning instructions are separate from the content".

Claim 2 recites the limitation "coupled provisioning instructions". There is insufficient antecedent basis for this limitation in the claim.

Similar rejections are applied to independent claim 21.

Claims 2-8, 10-20, 22-28, and 30-40 are also rejected based on virtue of their dependencies on the rejected base claims 1 and 21.

Claim Rejections – 35 USC §103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1-8, 10-28, and 30-42 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 2002/0131404 A1 to Mehta et al. (hereafter “Mehta”) in view of Kjellberg (US Patent No. 2003/0084165 A1) and Krantz (US Patent No. 2005/0091357 A1).

Claim 1:

Mehta discloses *a method for providing customized provisioning of an application on a runtime environment of a terminal* (FIG. 3, returning a suitable/customized provisioned application to a subscriber/terminal, 0075, 0113, 0136),

the application including content (FIG. 19 and related text, 0005, 0007, 0011, 0059, 0061-0064),

having at least one specified content type (FIG. 19, 20, and related text, application profile, device, URL, 0005, 0007, 0011, 0059, 0061-0064), *the method comprising the steps of:*

for each content type, obtaining the content by the runtime environment (e.g., FIG. 19, 20, and related text, obtaining application profile, device profile, and URL, 0010, 0064, 0090, 0092, 0097, 0113, 0114)*;*

obtaining by the runtime environment a set of provisioning instructions related to the content type (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application, 0008, 0010, 0075, 0092, 0136)*,*

the provisioning instructions being customized (FIG. 19, 20 and related text, the provisioning code/instructions are suitable/customized for each application, device, 0010, 0059, 0061-0067, 0075, 0077-0084)

by distributed provisioning control through the provisioning instructions (FIG. 5-7, Provisioning Manager 504 and related text, 0005, 0007, 0011, 0059, 0061-0064)*,*

the provisioning instructions embedded within content of the application for specifying a provisioning Application Program Interface (API) set for provisioning the content on the terminal (0010, 0064, 0090, 0092, 0097, 0113, 0114)*; and*

executing by the runtime environment the provisioning instructions for employing the API set to provision the application according to the specified content type (0010, 0064, 0090, 0092, 0097, 0113, 0114)*.*

Mehta does not explicitly disclose *[the provisioning instructions being customized] for different subsets of versions of the application.*

However, in an analogous art, Kjellbert further discloses *[the provisioning instructions being customized] for different subsets of versions of the application* (e.g., [0024]-[0026],

> "With reference now to FIG. 1 of the drawings, there is illustrated a provisioning server 200 capable of provisioning objects and applications to client devices 100 in real-time. As noted above, provisioning is the capability to receive a request for an application or object, find a suitable version of the requested application or object and provide the application or object to the

> requester. The ability to find a suitable version of the requested application or object accounts for the different formats utilized by the many different types of client devices 100, each with its own characteristics, limitations and configuration. For example, the client devices 100 may include PDAs 100a, workstations and desktop computers 100b, mobile phones 100c and laptops 100d. The characteristics and configurations of each of the different types of client devices 100 are stored in a device profiles database 230 within the provisioning server 200" (0025, an application has a plurality of versions based on different formats of client devices, emphasis added).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Kjellberg's teaching into Mehta's teaching. One would have been motivated to do so to provision a suitable/new version to client devices in real-time as suggested by Kjellberg (e.g., [0025]-[0026]).

Neither Mehta nor Kjellberg explicitly discloses *[executing by the runtime environment the provisioning instructions for employing the API set,] by a script interpreter*.

However, in an analogous art, Krantz further discloses *[executing by the runtime environment the provisioning instructions for employing the API set,] by a script interpreter* (interpreting XML-format files/XML provisioning files) as follows:

> [0047] Network provisioning services are potentially available via a variety of media (e.g., WWAN, dial-up, DSL, etc.). A user, through selection rule type "3" recited above (network provider service preference order), potentially specifies a rule that defers selection of a particular media to provisioning service-supplied rules. In an illustrative embodiment, the provisioning service 314 specifies such rules in the form of XML files.

> [0063] The following three methods are supported by the common RPC API 303 of the rules engine for the provisioning services 214. A CreateNetworkConfiguration method 434 is similar to the above-described SetNetworkAuthData method 432; however, a network configuration is supplied in XML format via an input parameter. An UpdateNetworkConfigurat- ion method 436 is similar to the CreateNetworkConfiguration method 434 except that a pre-existing configuration is assumed to exist for the identified network. A CreateUserAuthData method 438 is similar to the SetUserAuthData method 428 except the user data is supplied in a passed parameter in XML format.
>
> [0089] In yet another scenario, provided in the fourth row of the table, a rule applied by the rules engine 300 specifies a preference order comprising logical networks. A specific scenario example includes specifying: a corporate network over WISP A over WISP B. A network provider decides which physical network (WLAN over WWAN) to use based upon XML provisioning. Parameters used in such a rule scenario include: XML provisioning files from a wireless provider, business logic to facilitate dynamic network and interface selection. (emphasis added).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Krantz's teaching into Mehta and Kiellberg's teaching. One would have been motivated to do so to provide network provisioning services by using XML rules files, configuration files, and provisioning files as suggested by Krantz (e.g., [0047], [0063], and [0089]).

Claim 2:

Mehta discloses *the method according to claim 1, wherein provisioning control of the content is shared between the runtime environment and the application through the coupled provisioning instructions* (FIG. 19, 20, and related text, application profile, device, URL).

Claim 3:

Mehta discloses *the method according to claim 2 further comprising the step of employing a provisioning service to direct the provisioning API, the service configured for recognizing the provisioning instructions* (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application).

Claim 4:

Mehta discloses *the method according to claim 3 further comprising the step of the service customizing the provisioning process and the associated provisioning API set according to the provisioning instructions* (0005, 0007, 0011, 0059, 0061-0064).

Claim 5:

Mehta discloses *the method according to claim 4, wherein the service is shared by a plurality of the applications* (e.g., 0010, 0064, 0090, 0092, 0097, 0113, 0114).

Claim 6:

Mehta discloses *the method according to claim 3 further comprising the step of employing a standard one of the provisioning API set by the service* (e.g., 0008, 0010, 0075, 0092, 0136).

Claim 7:

Mehta discloses *the method according to claim 6 further comprising the step of obtaining remotely a custom API via a network coupled to the terminal* (0075, 0113, 0136).

Claim 8:

Mehta discloses *the method according to claim 2, wherein the provisioning instructions are selected from the group comprising code, script, and configuration data* (FIG. 19, 20, and related text, application profile, device, URL).

Claim 10:

Mehta discloses *the method according to claim 8, wherein the provisioning instructions are separate from the content* (0010, 0064, 0090, 0092, 0097, 0113, 0114).

Claim 11:

Mehta discloses *the method according to claim 10 further comprising the step of accessing the provisioning instructions remotely from the terminal* (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application).

Claim 12:

Mehta discloses *the method according to claim 11, wherein the remote access of the provisioning instructions is in conjunction with a networked repository* (0075, 0113, 0136).

Claim 13:

Mehta discloses *the method according to claim 12, wherein the terminal is selected from the group comprising wired devices and wireless devices* (0010, 0064, 0090, 0092, 0097, 0113, 0114).

Claim 14:

Mehta discloses *the method according to claim 5, wherein a generic API is included in the provisioning API set, the generic API configured for addressing by at least two dissimilar ones of the specified content type* (0005, 0007, 0011, 0059, 0061-0064).

Claim 15:

Mehta discloses *the method according to claim 14 further comprising the step of employing a series of enablers for providing access to corresponding selected ones of the generic API, each of the enablers associated with a predefined content type* (FIG. 19, 20, and related text, application profile, device, URL).

Claim 16:

Mehta discloses *the method according to claim 2, wherein a generic API is included in the provisioning API set, the generic API configured for addressing by at least two dissimilar ones of the specified content type* (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application).

Claim 17:

Mehta discloses *the method according to claim 16 further comprising the step of employing a series of enablers for providing access to corresponding selected ones of the generic API, each of the enablers associated with a predefined content type* (0010, 0064, 0090, 0092, 0097, 0113, 0114).

Claim 18:

Mehta discloses *the method according to claim 17, wherein the enabler is an executable unit that executes provisioning instruction requests for the predefined content type* (0075, 0113, 0136).

Claim 19:

Mehta discloses *the method according to claim 18 further comprising the step of obtaining the enabler selected from the group comprising: locally on the terminal by a provisioning service* (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application);

bundled with a content descriptor of the content; and remotely from the terminal by the provisioning service (0010, 0064, 0090, 0092, 0097, 0113, 0114).

Claim 20:

Mehta discloses *the method according to claim 5, wherein the provisioning instructions were amended prior to the step of obtaining the provisioning instructions by the runtime environment* (FIG. 19, 20, and related text, application profile, device, URL).

Claim 21:

Mehta discloses *a terminal, including a computer processor and a computer readable storage medium for providing customized provisioning of an application on a runtime environment* (FIG. 3, returning a suitable/customized provisioned application to a subscriber/terminal, 0075, 0113, 0136),

the application including content (FIG. 19, 20, and related text, application profile, device, URL, 0005, 0007, 0011, 0059, 0061-0064)

having at least one specified content type (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application, 0008, 0010, 0075, 0092, 0136), *the terminal comprising:*

a processing framework for obtaining the content (FIG. 19, 20, and related text, obtaining application profile, device profile, and URL, 0010, 0064, 0090, 0092, 0097, 0113, 0114);

obtaining by the runtime environment a set of provisioning instructions related to the content type (FIG. 19, 20, and related text, obtaining application profile, device profile, and URL, 0010, 0064, 0090, 0092, 0097, 0113, 0114),

the provisioning instructions being customized (FIG. 19 and related text, 0005, 0007, 0011, 0059, 0061-0064)

by distributed provisioning control through the provisioning instructions (e.g., FIG. 2, execution/control of provisioning instructions/APIs have been distributed over Provisioning Application 208, Provisioning API 222, Adapter 206, but not been hardcoded in target devices 202),

a provisioning API set included in the processing framework for provisioning the content (FIG. 19, 20 and related text, the provisioning code/instructions are suitable/customized for each application, device, 0010, 0059, 0061-0067, 0075, 0077-0084)*; and*

a set of provisioning instructions related to the content, the provisioning instructions embedded in content of the application for specifying selected ones of the provisioning API set (FIG. 3, returning a suitable/customized provisioned application to a subscriber/terminal, 0075, 0113, 0136)*.*

Mehta does not explicitly disclose *the provisioning instructions being customized for different subsets of versions of the application.*

However, in an analogous art, Kjellbert further discloses *[the provisioning instructions being customized] for different subsets of versions of the application* (e.g., [0024]-[0026],

> "With reference now to FIG. 1 of the drawings, there is illustrated a provisioning server 200 capable of provisioning objects and applications to client devices 100 in real-time. As noted above, provisioning is the capability to receive a request for an application or object, find a suitable version of the requested application or object and provide the application or object to the requester. The ability to find a suitable version of the requested application or object accounts for the different formats utilized by the many different types of client devices 100, each with its own characteristics, limitations and configuration. For example, the client devices 100 may include PDAs 100a, workstations and desktop computers 100b, mobile phones 100c and laptops 100d. The characteristics and configurations of each of the different types of client devices 100 are stored in a device profiles database 230 within the provisioning server 200" (0025, an application

> has a plurality of versions based on different formats of client devices, emphasis added).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Kjellberg's teaching into Mehta's teaching. One would have been motivated to do so to provision a suitable/new version to client devices in real-time as suggested by Kjellberg (e.g., [0025]-[0026]).

Neither Mehta nor Kjellberg explicitly discloses *executing by the runtime environment the provisioning instructions for employing the API set, by a script interpreter*.

However, in an analogous art, Krantz further discloses *[executing by the runtime environment the provisioning instructions for employing the API set,] by a script interpreter* (interpreting XML-format files/XML provisioning files) as follows:

> [0047] Network provisioning services are potentially available via a variety of media (e.g., WWAN, dial-up, DSL, etc.). A user, through selection rule type "3" recited above (network provider service preference order), potentially specifies a rule that defers selection of a particular media to provisioning service-supplied rules. In an illustrative embodiment, the provisioning service 314 specifies such rules in the form of XML files.
>
> [0063] The following three methods are supported by the common RPC API 303 of the rules engine for the provisioning services 214. A CreateNetworkConfiguration method 434 is similar to the above-described SetNetworkAuthData method 432; however, a network configuration is supplied in XML format via an input parameter. An UpdateNetworkConfigurat- ion method 436 is similar to the CreateNetworkConfiguration method 434 except that a pre-existing configuration is assumed to exist

> for the identified network. A CreateUserAuthData method 438 is similar to the SetUserAuthData method 428 except the user data is supplied in a passed parameter in XML format.
>
> [0089] In yet another scenario, provided in the fourth row of the table, a rule applied by the rules engine 300 specifies a preference order comprising logical networks. A specific scenario example includes specifying: a corporate network over WISP A over WISP B. A network provider decides which physical network (WLAN over WWAN) to use based upon XML provisioning. Parameters used in such a rule scenario include: XML provisioning files from a wireless provider, business logic to facilitate dynamic network and interface selection. (emphasis added).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Krantz's teaching into Mehta and Kiellberg's teaching. One would have been motivated to do so to provide network provisioning services by using XML rules files, configuration files, and provisioning files as suggested by Krantz (e.g., [0047], [0063], and [0089]).

Claim 22:

Mehta discloses *the terminal according to claim 21, wherein provisioning control of the content is shared between the framework and the application through the coupled provisioning instructions* (FIG. 19, 20, and related text, application profile, device, URL).

Claim 23:

Mehta discloses *the terminal according to claim 22 further comprising a provisioning service to direct the provisioning API, the service configured for recognizing the provisioning instructions* (e.g., 0010, 0064, 0090, 0092, 0097, 0113, 0114).

Claim 24:

Mehta discloses *the terminal according to claim 23 wherein the service is configured for customizing the provisioning process and the associated provisioning API set according to the provisioning instructions* (0005, 0007, 0011, 0059, 0061-0064).

Claim 25:

Mehta discloses *the terminal according to claim 24, wherein the service is shared by a plurality of the applications* (0008, 0010, 0075, 0092, 0136).

Claim 26:

Mehta discloses *the terminal according to claim 23, wherein the service employs a standard one of the provisioning API set* (0010, 0064, 0090, 0092, 0097, 0113, 0114);

Claim 27:

Mehta discloses *the terminal according to claim 26, a custom API is obtained remotely by the service via a network coupled to the terminal* (0075, 0113, 0136).

Claim 28:

Mehta discloses *the terminal according to claim 22, wherein the provisioning instructions are selected from the group comprising code, script, and configuration data* (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application).

Claim 30:

Mehta discloses *the terminal according to claim 28, wherein the provisioning instructions are separate from the content* (FIG. 19, 20, and related text, application profile, device, URL).

Claim 31:

Mehta discloses *the terminal according to claim 30, wherein the provisioning instructions are configured for obtaining the remotely from the terminal* (0010, 0064, 0090, 0092, 0097, 0113, 0114).

Claim 32:

Mehta discloses *the terminal according to claim 31, wherein the remote access of the provisioning instructions is in conjunction with a networked repository* (0005, 0007, 0011, 0059, 0061-0064).

Claim 33:

Mehta discloses *the terminal according to claim 32, wherein the terminal is selected from the group comprising wired devices and wireless devices* (0075, 0113, 0136).

Claim 34:

Mehta discloses *the terminal according to claim 25, wherein a generic API is included in the provisioning API set, the generic API configured for addressing by at least two dissimilar ones of the specified content type* (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application).

Claim 35:

Mehta discloses *the terminal according to claim 34 further comprising a series of enablers for providing access to corresponding selected ones of the generic API, each of the enablers associated with a predefined content type* (FIG. 19, 20, and related text, application profile, device, URL).

Claim 36:

Mehta discloses *the terminal according to claim 22, wherein a generic API is included in the provisioning API set, the generic API configured for addressing by at*

least two dissimilar ones of the specified content type (0010, 0064, 0090, 0092, 0097, 0113, 0114).

Claim 37:

Mehta discloses *the terminal according to claim 36 further comprising a series of enablers for providing access to corresponding selected ones of the generic API, each of the enablers associated with a predefined content type* (0005, 0007, 0011, 0059, 0061-0064).

Claim 38:

Mehta discloses *the terminal according to claim 37, wherein the enabler is an executable unit that executes provisioning instruction requests for the predefined content type* (0008, 0010, 0075, 0092, 0136).

Claim 39:

Mehta discloses *the terminal according to claim 38, wherein the enabler location is selected from the group comprising: locally on the terminal by a provisioning service* (FIG. 19, 20, and related text, application profile, device, URL)*;*

bundled with a content descriptor of the content; and remotely from the terminal by the provisioning service (0075, 0113, 0136).

Claim 40:

Mehta discloses *the terminal according to claim 25, wherein the provisioning instructions were amended prior to the step of obtaining the provisioning instructions by the runtime environment* (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application).

Claim 41:

Mehta discloses *a method for providing customized provisioning of an application on a runtime environment of a terminal* (FIG. 19 and related text, 0005, 0007, 0011, 0059, 0061-0064)*,*

the application including content (FIG. 3, returning a suitable/customized provisioned application to a subscriber/terminal, 0075, 0113, 0136)

having at least one specified content type (FIG. 19, 20, and related text, application profile, device, URL, 0005, 0007, 0011, 0059, 0061-0064)*, the method comprising the steps of:*

sending the content for receipt by the runtime environment (FIG. 19, 20, and related text, obtaining application profile, device profile, and URL, 0010, 0064, 0090, 0092, 0097, 0113, 0114)*;*

obtaining by the runtime environment a set of provisioning instructions related to the content type (FIG. 19, 20 and related text, the provisioning code/instructions are suitable/customized for each application, device, 0010, 0059, 0061-0067, 0075, 0077-0084)*,*

the provisioning instructions being customized (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application, 0008, 0010, 0075, 0092, 0136)

by distributed provisioning control through the provisioning instructions (FIG. 19 and related text, 0005, 0007, 0011, 0059, 0061-0064)*,*

sending a set of provisioning instructions related to the content for receipt by the runtime environment, the provisioning instructions embedded in content of the application for specifying a provisioning API set for provisioning the content (FIG. 3, returning a suitable/customized provisioned application to a subscriber/terminal, 0075, 0113, 0136)*; and*

making available selected ones of the API provisioning set for use by the provisioning instructions; wherein execution of the provisioning instructions employs the API provisioning set to provision the application according to the specified content type (FIG. 5-7, Provisioning Manager 504 and related text, 0005, 0007, 0011, 0059, 0061-0064).

Mehta does not explicitly disclose *the provisioning instructions being customized for different subsets of versions of the application.*

However, in an analogous art, Kjellbert further discloses *[the provisioning instructions being customized] for different subsets of versions of the application* (e.g., [0024]-[0026],

> "With reference now to FIG. 1 of the drawings, there is illustrated a provisioning server 200 capable of provisioning objects and applications to client devices 100 in real-time. As noted above, provisioning is the capability to receive a request for an application or object, find a suitable version of the requested application or object and provide the application or object to the requester. The ability to find a suitable version of the requested application or object accounts for the different formats utilized by the many different types of client devices 100, each with its own characteristics, limitations and configuration. For example, the client devices 100 may include PDAs 100a, workstations and desktop computers 100b, mobile phones 100c and laptops 100d. The characteristics and configurations of each of the different types of client devices 100 are stored in a device profiles database 230 within the provisioning server 200" (0025, an application has a plurality of versions based on different formats of client devices, emphasis added).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Kjellberg's teaching into Mehta's teaching. One would have been motivated to do so to provision a suitable/new version to client devices in real-time as suggested by Kjellberg (e.g., [0025]-[0026]).

Neither Mehta nor Kjellberg explicitly discloses *executing by the runtime environment the provisioning instructions for employing the API set, by a script interpreter*.

However, in an analogous art, Krantz further discloses *[executing by the runtime environment the provisioning instructions for employing the API set,] by a script interpreter* (interpreting XML-format files/XML provisioning files) as follows:

> [0047] Network provisioning services are potentially available via a variety of media (e.g., WWAN, dial-up, DSL, etc.). A user, through selection rule type "3" recited above (network provider service preference order), potentially specifies a rule that defers selection of a particular media to provisioning service-supplied rules. In an illustrative embodiment, the provisioning service 314 specifies such rules in the form of XML files.
>
> [0063] The following three methods are supported by the common RPC API 303 of the rules engine for the provisioning services 214. A CreateNetworkConfiguration method 434 is similar to the above-described SetNetworkAuthData method 432; however, a network configuration is supplied in XML format via an input parameter. An UpdateNetworkConfigurat- ion method 436 is similar to the CreateNetworkConfiguration method 434 except that a pre-existing configuration is assumed to exist for the identified network. A CreateUserAuthData method 438 is similar to the SetUserAuthData method 428 except the user data is supplied in a passed parameter in XML format.
>
> [0089] In yet another scenario, provided in the fourth row of the table, a rule applied by the rules engine 300 specifies a preference order comprising logical networks. A

> specific scenario example includes specifying: a corporate network over WISP A over WISP B. A network provider decides which physical network (WLAN over WWAN) to use based upon XML provisioning. Parameters used in such a rule scenario include: XML provisioning files from a wireless provider, business logic to facilitate dynamic network and interface selection. (emphasis added).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Krantz's teaching into Mehta and Kiellberg's teaching. One would have been motivated to do so to provide network provisioning services by using XML rules files, configuration files, and provisioning files as suggested by Krantz (e.g., [0047], [0063], and [0089]).

Claim 42:

Mehta discloses *a computer program product for providing customized provisioning of an application on a runtime environment of a terminal* (FIG. 19 and related text, 0005, 0007, 0011, 0059, 0061-0064),

the application including content (FIG. 19, 20, and related text, application profile, device, URL, 0005, 0007, 0011, 0059, 0061-0064)

having at least one specified content type (FIG. 3, returning a suitable/customized provisioned application to a subscriber/terminal, 0075, 0113, 0136), *the computer program product comprising:*

a computer readable medium; a processing framework module stored on the computer readable medium for obtaining the content (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application, 0008, 0010, 0075, 0092, 0136);

a provisioning service module coupled to the framework module, the provisioning service module configured for utilizing a provisioning API set for provisioning the content (FIG. 19, 20, and related text, obtaining application profile, device profile, and URL, 0010, 0064, 0090, 0092, 0097, 0113, 0114);

obtaining by the runtime environment a set of provisioning instructions related to the content type (FIG. 19, 20 and related text, the provisioning code/instructions are suitable/customized for each application, device, 0010, 0059, 0061-0067, 0075, 0077-0084)*,*

the provisioning instructions being customized (FIG. 19, 20, and related text, application profile, device, URL, 0005, 0007, 0011, 0059, 0061-0064)

by distributed provisioning control through the provisioning instructions (FIG. 23, 26, and related text, provisioning code/instructions have been instrumented in the provisioned application, 0008, 0010, 0075, 0092, 0136)*, and*

an interpreter module coupled to the framework module, the interpreter module configured for executing a set of provisioning instructions related to the content (FIG. 5-7, Provisioning Manager 504 and related text, 0005, 0007, 0011, 0059, 0061-0064)*,*

the provisioning instructions embedded in content of the application for specifying selected ones of the provisioning API set (0010, 0064, 0090, 0092, 0097, 0113, 0114)*.*

Mehta does not explicitly disclose *the provisioning instructions being customized for different subsets of versions of the application.*

However, in an analogous art, Kjellbert further discloses *[the provisioning instructions being customized] for different subsets of versions of the application* (e.g., [0024]-[0026],

> "With reference now to FIG. 1 of the drawings, there is illustrated a provisioning server 200 capable of provisioning objects and applications to client devices 100 in real-time. As noted above, provisioning is the capability to receive a request for an application or object, find a suitable version of the requested application or object and provide the application or object to the requester. The ability to find a suitable version of the requested application or object accounts for the different formats utilized by

> the many different types of client devices 100, each with its own characteristics, limitations and configuration. For example, the client devices 100 may include PDAs 100a, workstations and desktop computers 100b, mobile phones 100c and laptops 100d. The characteristics and configurations of each of the different types of client devices 100 are stored in a device profiles database 230 within the provisioning server 200" (0025, an application has a plurality of versions based on different formats of client devices, emphasis added).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Kjellberg's teaching into Mehta's teaching. One would have been motivated to do so to provision a suitable/new version to client devices in real-time as suggested by Kjellberg (e.g., [0025]-[0026]).

Neither Mehta nor Kjellberg explicitly discloses *executing by the runtime environment the provisioning instructions for employing the API set, by a script interpreter*.

However, in an analogous art, Krantz further discloses *executing by the runtime environment the provisioning instructions for employing the API set, by a script interpreter* (e.g., [0089], XML parser/interpreter as a script interpreter).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Krantz's teaching into Mehta and Kiellberg's teaching. One would have been motivated to do so to provide network provisioning services by using XML rules files, configuration files, and provisioning files as suggested by Krantz (e.g., [0047], [0063], and [0089]).

Conclusion

9. Any inquiry concerning this communication should be directed to examiner Thuy (Twee) Dao, whose telephone/fax numbers are (571) 272 8570 and (571) 273 8570,

respectively. The examiner can normally be reached on every Tuesday, Thursday, and Friday from 6:00AM to 6:00PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam, can be reached at (571) 272 3695.

Any inquiry of a general nature of relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is (571) 272 2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Thuy Dao/ (Twee)
Examiner, Art Unit 2192